# Sentiment Analysis on Amazon Fine Food Reviews

**Mt. SAC CISB 62 Final Project - Fall 2023**

**Russell Luna**

## https://github.com/rmoon64/CISB62_Final/ (https://github.com/rmoon64/CISB62_Final/)

# Introduction / Summary:

### The 'Amazon Fine Food Reviews' from Kaggle is a complete dataset that has about 560,000 reviews, which we trim down to 50,000 reviews. The goal of the project is to recommend new products to someone who does reviews often (Chris), using RNN to perform sentimental analysis using only Amazon reviews as the model.

### This project utilized various deep learning techniques. The main tools used were RNN, TensorFlow V1, and hyperparameter tuning. First, I utilized a trimmed-down version of the data set with 50,000 reviews. Then, I used Matplotlib to plot the visualizations of important data, for example, the top 10 products with the most reviewers and the percentage of positive and negative reviews made by Chris. I then prepared the RNN with the LSTM model. This model was then fitted using six epochs and a batch size of twenty. This proved to be very beneficial, as the accuracy was extremely high at 0.9953 with a quite low loss of 0.0160. Next, I cleaned the data and defined the parameters and placeholders. Lastly, I got the scores and recommendations for Chris.

https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews (https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews)

In [1]:

```python
import tensorflow as tf
import seaborn as sns
import re
import pandas as pd
import numpy as np
import keras
import datetime

from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Embedding
from tensorflow.keras.callbacks import ModelCheckpoint
from string import punctuation
from sklearn.model_selection import train_test_split
from nltk.corpus import stopwords
from matplotlib import pyplot as plt
from keras.preprocessing.text import Tokenizer
```

```
WARNING:tensorflow:From C:\Anaconda\Lib\site-packages\keras\src\losses.p
y:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. P
lease use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [2]:

```python
import warnings
warnings.filterwarnings("ignore")
```

In [3]:

```python
#Import dataset
df = pd.read_csv('Reviews.csv')
```

In [4]:

```python
df.describe()
```

Out[4]:

| | Id | HelpfulnessNumerator | HelpfulnessDenominator | Score | |
|---|---|---|---|---|---|
| **count** | 50000.000000 | 50000.000000 | 50000.000000 | 50000.000000 | 5.00000 |
| **mean** | 25000.500000 | 1.603360 | 2.060260 | 4.145840 | 1.29519 |
| **std** | 14433.901067 | 5.620771 | 6.216044 | 1.325596 | 4.73462 |
| **min** | 1.000000 | 0.000000 | 0.000000 | 1.000000 | 9.61718 |
| **25%** | 12500.750000 | 0.000000 | 0.000000 | 4.000000 | 1.26964 |
| **50%** | 25000.500000 | 0.000000 | 1.000000 | 5.000000 | 1.30913 |
| **75%** | 37500.250000 | 2.000000 | 2.000000 | 5.000000 | 1.33107 |
| **max** | 50000.000000 | 398.000000 | 401.000000 | 5.000000 | 1.35121 |

In [5]:
```
df.head(3)
```

Out[5]:

| | Id | ProductId | UserId | ProfileName | HelpfulnessNumerator | HelpfulnessD |
|---|---|---|---|---|---|---|
| **0** | 1 | B001E4KFG0 | A3SGXH7AUHU8GW | delmartian | 1 | |
| **1** | 2 | B00813GRG4 | A1D87F6ZCVE5NK | dll pa | 0 | |
| **2** | 3 | B000LQOCH0 | ABXLMWJIXXAIN | Natalia Corres "Natalia Corres" | 1 | |

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Id                      50000 non-null  int64 
 1   ProductId               50000 non-null  object
 2   UserId                  50000 non-null  object
 3   ProfileName             49995 non-null  object
 4   HelpfulnessNumerator    50000 non-null  int64 
 5   HelpfulnessDenominator  50000 non-null  int64 
 6   Score                   50000 non-null  int64 
 7   Time                    50000 non-null  int64 
 8   Summary                 49998 non-null  object
 9   Text                    50000 non-null  object
dtypes: int64(5), object(5)
memory usage: 3.8+ MB
```

In [7]:
```
print(df.shape)
```

```
(50000, 10)
```

In [8]:
```
df.head()
```

Out[8]:

| | Id | ProductId | UserId | ProfileName | HelpfulnessNumerator | HelpfulnessD |
|---|---|---|---|---|---|---|
| 0 | 1 | B001E4KFG0 | A3SGXH7AUHU8GW | delmartian | 1 | |
| 1 | 2 | B00813GRG4 | A1D87F6ZCVE5NK | dll pa | 0 | |
| 2 | 3 | B000LQOCH0 | ABXLMWJIXXAIN | Natalia Corres "Natalia Corres" | 1 | |
| 3 | 4 | B000UA0QIQ | A395BORC6FGVXV | Karl | 3 | |
| 4 | 5 | B006K2ZZ7K | A1UQRSCLF8GW1T | Michael D. Bigham "M. Wassir" | 0 | |

In [9]:
```
df["Text"][0]
```

Out[9]:
```
'I have bought several of the Vitality canned dog food products and have
found them all to be of good quality. The product looks more like a stew
than a processed meat and it smells better. My Labrador is finicky and s
he appreciates this product better than  most.'
```

In [10]:
```
#Grabbing the columns that we need
df_reviews = df[['ProductId','ProfileName','Score','Text','HelpfulnessNume
```

In [11]:
```
#Loading the top 10 reviewers with the most reviews.
df.pivot_table(columns=['ProfileName'], aggfunc='size').sort_values(ascend
```

Out[11]:
```
ProfileName
Gary Peterson                               44
C. F. Hill "CFH"                            39
O. Brown "Ms. O. Khannah-Brown"             35
c2                                          35
Gunner                                      31
Chris                                       31
Rebecca of Amazon "The Rebecca Review"      29
Laura                                       24
Amanda                                      24
Mike                                        24
dtype: int64
```

In [12]:
```
popular = df[['ProductId','ProfileName']].groupby('ProductId').count().sor
```

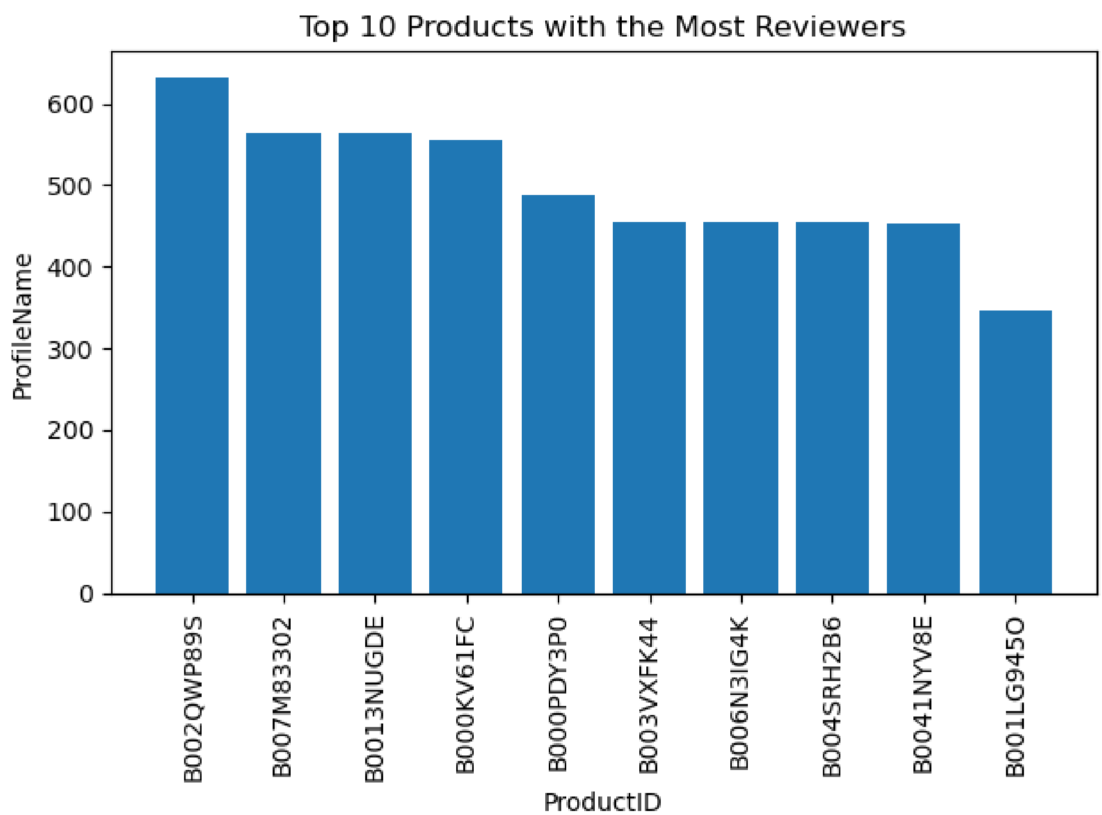
In [13]:
```
df.columns
```

Out[13]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerato
r',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

## Visuals: matplotlib and seaborn

In [14]:
```
top_10_popular = popular.head(10) # Select the top 10 products with the mo
```

In [15]:
```
fig, axs = plt.subplots(sharey=True, tight_layout=True)
axs.bar(top_10_popular['ProductId'], top_10_popular['ProfileName'])
axs.set_xlabel('ProductID')
axs.set_ylabel('ProfileName')
axs.set_title('Top 10 Products with the Most Reviewers')
plt.xticks(rotation=90) # Rotate x-axis labels for better visibility
plt.show()
```



In [16]:

```
df_reviews[df_reviews['ProfileName']=='Chris']
```

Out[16]:

| | ProductId | ProfileName | Score | Text | Help |
|---|---|---|---|---|---|
| **151** | B00374XSVY | Chris | 5 | Works with chicken fish beef or pork. Fast eas... | |
| **637** | B000G6RYNE | Chris | 4 | This kettle chips taste "Good , Crispy & Crunc... | |
| **729** | B008BEGP9W | Chris | 4 | I really like the pineapple shortcakes sold he... | |
| **1006** | B002XG21MO | Chris | 5 | These are just like the animal crackers we eat... | |
| **3142** | B000FDKQCY | Chris | 5 | This product is as good as any that I have eve... | |
| **5301** | B003OJLCXI | Chris | 4 | I really like this tea, but my husband smelled... | |
| **5826** | B000633O2C | Chris | 5 | I want to give my cats as much variety as poss... | |
| **7424** | B005LMLXN0 | Chris | 5 | As a tea-junkie, Runa's traditional guayusa of... | |
| **7653** | B000TRFGGM | Chris | 4 | I opened the carton and the plant looked like ... | |
| **7812** | B00474H936 | Chris | 4 | I have tried almost every kind of "fake meat" ... | |
| **9712** | B001LXYA5Q | Chris | 4 | This stuff works, but be warned the effect doe... | |
| **14414** | B00063KO34 | Chris | 5 | My dogs love these chews. I have 5 dogs - all ... | |
| **15075** | B00113L7KC | Chris | 5 | I love this tea. I don't profess to be a tea ... | |
| **15150** | B0013E21V8 | Chris | 5 | Our family loves this syrup! I love that it i... | |
| **16240** | B007TJGZ54 | Chris | 5 | Love the Green Mountain Coffee; tastes great i... | |
| **16546** | B003P7WLF2 | Chris | 1 | I purchased these in lieu of the V8 Fusions si... | |
| **18674** | B00008JOL0 | Chris | 5 | Zuke's is the best and my dog loves to eat the... | |
| **21217** | B002QWP89S | Chris | 5 | Molly and Cocoa, our cocker spaniels, said to ... | |
| **21607** | B002QWP89S | Chris | 5 | I am happy with the service of this company. ... | |
| **26212** | B0058AMY74 | Chris | 4 | This kettle chips taste "Good , Crispy & Crunc... | |
| **26717** | B001IZG6P4 | Chris | 5 | These are the best peppercorns for the money I... | |
| **30041** | B000DZFMEQ | Chris | 5 | My wife is gluten and dairy intolerant and we ... | |
| **32735** | B0083QJU72 | Chris | 3 | Having grown up in sugar maple country and bou... | |
| **36181** | B000ILKXP8 | Chris | 3 | I've seen some reviewers commenting on the sug... | |
| **38414** | B002QK0RJG | Chris | 5 | <a href="http://www.amazon.com/gp/product/B004... | |
| **41755** | B0000D8DI0 | Chris | 4 | This is excellent chocolate. Hachez is a qual... | |
| **43455** | B001EQ4P2I | Chris | 5 | I'm a big fan of Blue Diamond brand Almonds. I... | |
| **43457** | B001EQ4P2I | Chris | 5 | I'm a big fan of the Blue Diamond brand almond... | |
| **43524** | B001EQ4P2I | Chris | 3 | To start, I'm a big fan of Blue Diamond brand ... | |
| **46281** | B001EO5KSS | Chris | 4 | These are great! Our whole family likes them.... | |
| **49031** | B000X1F9XY | Chris | 1 | 2 of my 3 dogs destroyed these in less than 30... | |

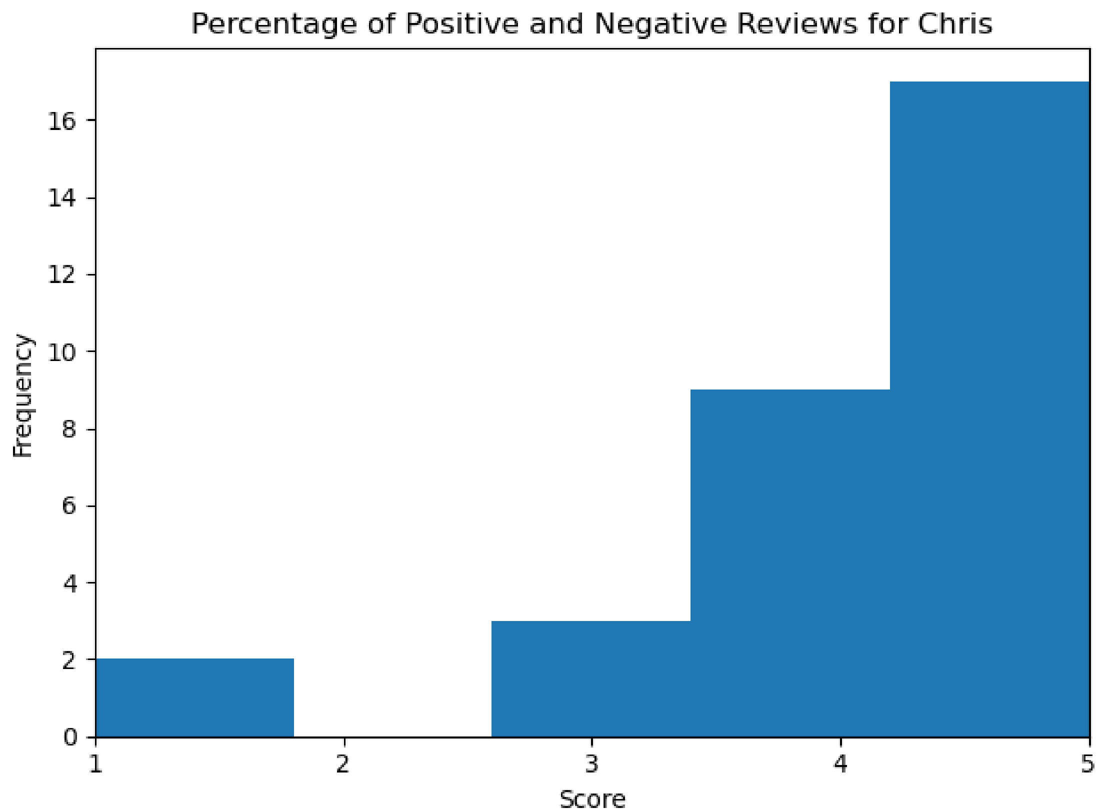
## Plotting Chris' percentage of positive views vs negative reviews

In [17]:

```
df_chris = df_reviews[df_reviews['ProfileName']=='Chris']
```

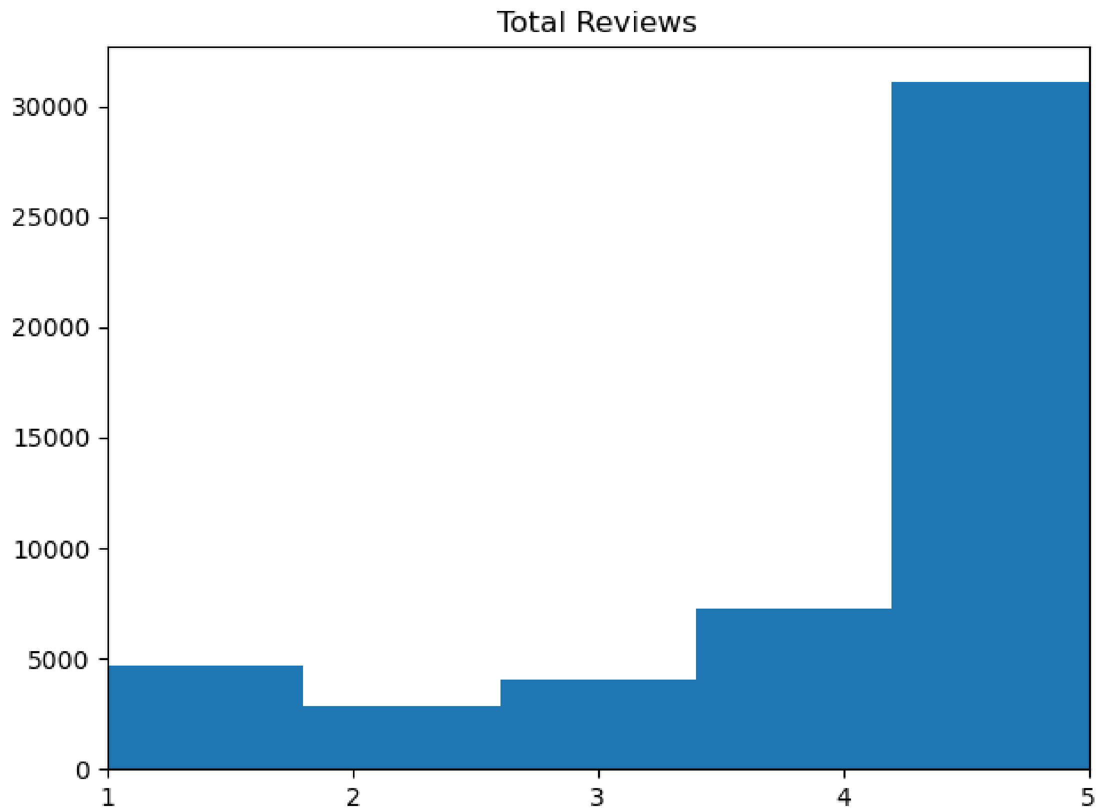
In [18]:

```
fig, axs = plt.subplots(sharey=True, tight_layout=True)
axs.hist(df_chris['Score'], bins=5)
plt.xlim([1, 5])
plt.xticks(np.arange(1, 6, 1))
axs.set_title('Percentage of Positive and Negative Reviews for Chris')
axs.set_xlabel('Score')
axs.set_ylabel('Frequency')
plt.show()
```



## Plotting the total reviews

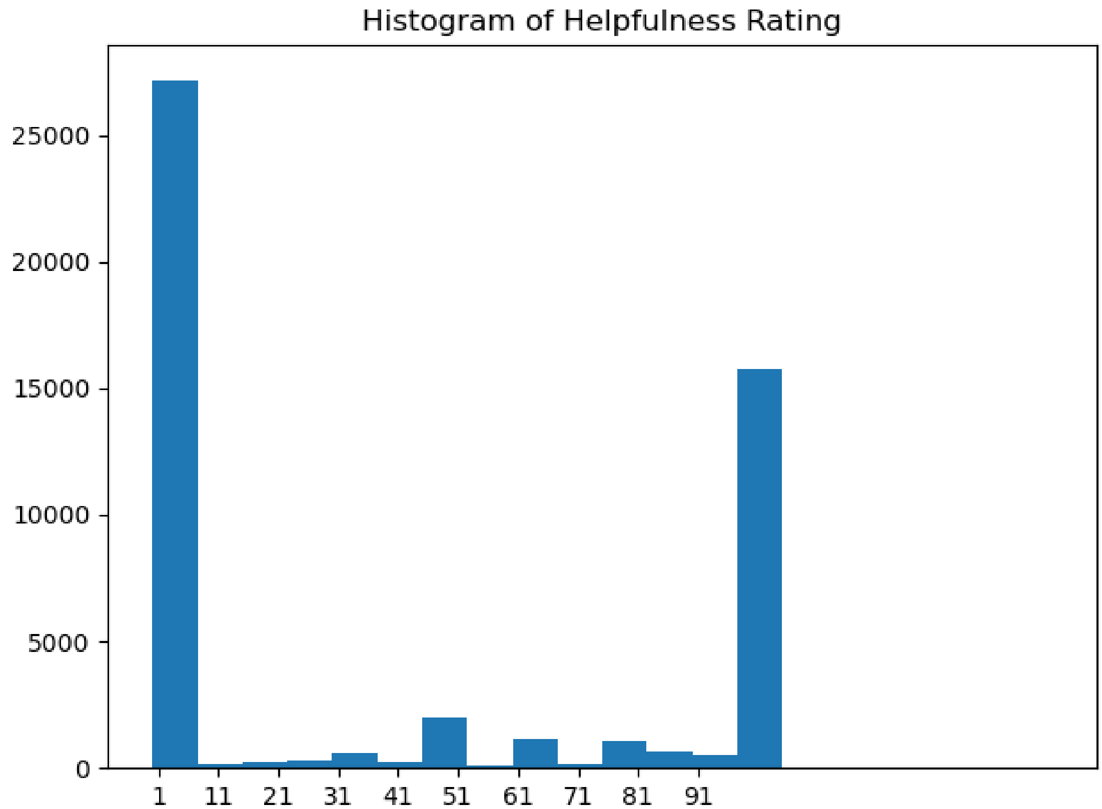
In [19]:
```
fig, axs = plt.subplots(sharey=True, tight_layout=True)
axs.hist(df_reviews['Score'],bins=5)
plt.xlim([1, 5])
plt.xticks(np.arange(1, 6, 1))
plt.title('Total Reviews')
plt.show()
```




## Histogram of helpfulness rating

In [20]:
```
df_reviews['Percent'] = (df_reviews['HelpfulnessNumerator'] / df_reviews['
df_reviews['Percent'].fillna(0,inplace=True)
```

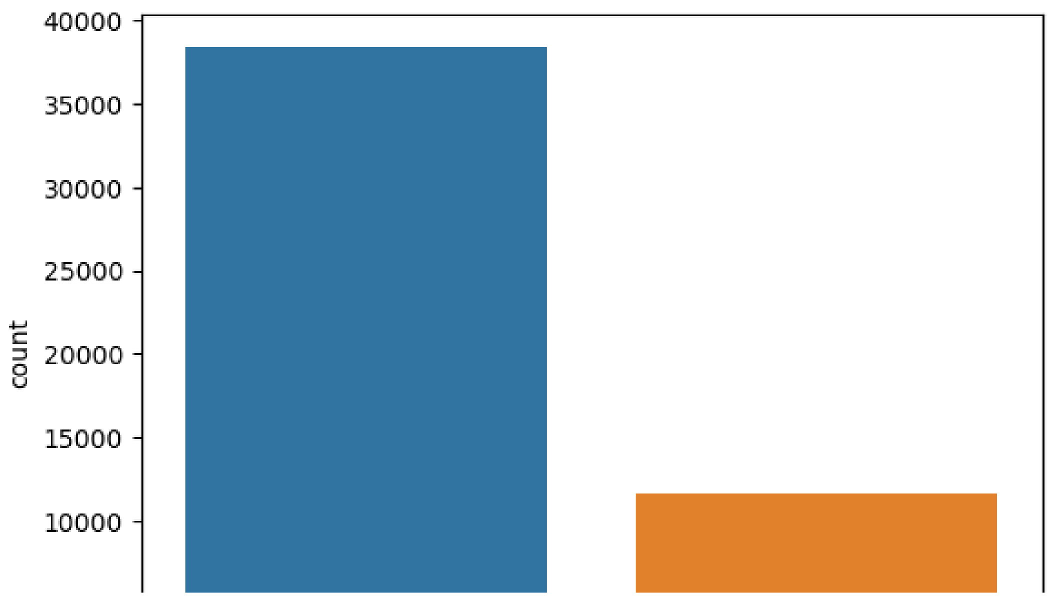
In [21]:
```
fig, axs = plt.subplots(sharey=True, tight_layout=True)
axs.hist(df_reviews['Percent'],bins=20)
plt.xticks(np.arange(1, 100, 10))
plt.title('Histogram of Helpfulness Rating')
plt.show()
```




In [22]:
```
df_reviews['Polarity_Rating'] = df_reviews['Score'].apply(lambda x: 'Posit
```

In [23]:
```python
sns.countplot(data = df_reviews, x= 'Polarity_Rating')
```

Out[23]: <Axes: xlabel='Polarity_Rating', ylabel='count'>




In [24]:
```python
df_reviews[df_reviews['Polarity_Rating'] == 'Negative'].shape[0]
```

Out[24]: 11582

In [25]:
```python
data_Positive = df_reviews[df_reviews['Polarity_Rating'] == 'Positive'][0:
data_Negative = df_reviews[df_reviews['Polarity_Rating'] == 'Negative']

data_Negative_over = data_Negative.sample(20000, replace=True)
df_balance_reviews = pd.concat([data_Positive, data_Negative_over], axis=0
```

In [26]:
```python
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\RL\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[26]: True

In [27]:
```python
#Setting the stop words
english_stops = set(stopwords.words('english'))
```

In [28]:

```
df = df_balance_reviews
x_data = df['Text']         # Reviews/Input
y_data = df['Polarity_Rating']     # Sentiment/Output

# PRE-PROCESS REVIEW
x_data = x_data.replace({'<.*?>': ''}, regex = True)          # remove htm
x_data = x_data.replace({'[^A-Za-z]': ' '}, regex = True)     # remove non
x_data = x_data.apply(lambda review: [w for w in review.split() if w not i
x_data = x_data.apply(lambda review: [w.lower() for w in review])    # Lowe

# ENCODE SENTIMENT -> 0 & 1
y_data = y_data.replace('Positive', 1)
### simmilarly for y_data
y_data = y_data.replace('Negative', 0)
```

In [29]:

```
X_train, X_test, y_train, y_test = train_test_split(x_data, y_data, test_s
```

In [30]:

```
def get_max_length(x_train):
    review_length = []
    for review in x_train:
        review_length.append(len(review))

    return int(np.ceil(np.mean(review_length)))
```

In [31]:

```
max_length = get_max_length(X_train)
print(max_length)
```

```
48
```

In [32]:

```
# ENCODE REVIEW
token = Tokenizer(lower=False)    # False becuase we already did it.
token.fit_on_texts(X_train)
x_train = token.texts_to_sequences(X_train)
x_test = token.texts_to_sequences(X_test)

### this is the code to add the pad to x_train
x_train = pad_sequences(x_train, maxlen=max_length, padding='post', trunca

############# add pad to x_test   ###########################
x_test = pad_sequences(x_test, maxlen=max_length, padding='post', truncati

total_words = len(token.word_index) + 1   # we need to add 1 because of 0
```

In [33]:

```
# ARCHITECTURE
EMBED_DIM = 3200
LSTM_OUT = 64

#### add the model here:
model = Sequential()

#model.add(Embedding(input_dim=max_length, output_dim=LSTM_OUT,input_lengt
model.add(Embedding(total_words, EMBED_DIM, input_length = max_length))
model.add(LSTM(LSTM_OUT))
model.add(Dense(units=1, activation='sigmoid'))

### compile the model using: optimizer = 'adam', loss = 'binary_crossentro
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accu

#Adding a Checkpoint to save the model.
checkpoint = ModelCheckpoint('models/LSTM.h5', monitor='accuracy', save_be
```

```
WARNING:tensorflow:From C:\Anaconda\Lib\site-packages\keras\src\backend.
py:873: The name tf.get_default_graph is deprecated. Please use tf.compa
t.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Anaconda\Lib\site-packages\keras\src\optimize
rs\__init__.py:309: The name tf.train.Optimizer is deprecated. Please us
e tf.compat.v1.train.Optimizer instead.
```

In [34]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 48, 3200)          84083200

 lstm (LSTM)                 (None, 64)                835840

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 84919105 (323.94 MB)
Trainable params: 84919105 (323.94 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

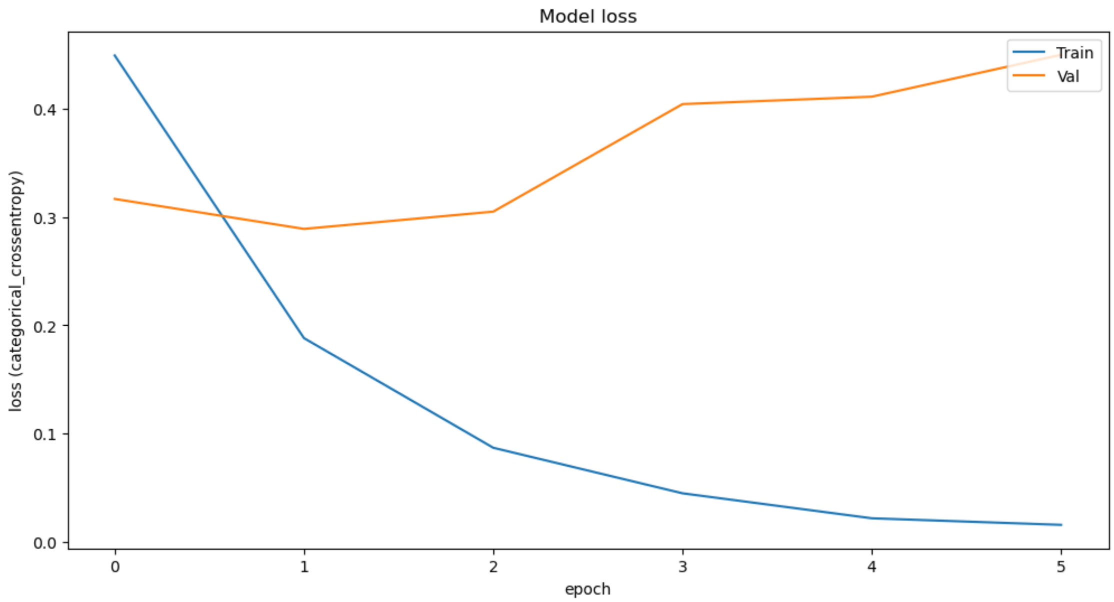
In [35]:
```
# Fit the model
history = model.fit(x_train, y_train, epochs=6, batch_size=20, callbacks=
```

```
Epoch 1/6
WARNING:tensorflow:From C:\Anaconda\Lib\site-packages\keras\src\utils\tf
_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Pleas
e use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Anaconda\Lib\site-packages\keras\src\engine\b
ase_layer_utils.py:384: The name tf.executing_eagerly_outside_functions
is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functio
ns instead.

1300/1300 [==============================] - ETA: 0s - loss: 0.4488 - ac
curacy: 0.7871
Epoch 1: accuracy improved from -inf to 0.78708, saving model to models
\LSTM.h5
1300/1300 [==============================] - 722s 550ms/step - loss: 0.4
488 - accuracy: 0.7871 - val_loss: 0.3165 - val_accuracy: 0.8706
Epoch 2/6
1300/1300 [==============================] - ETA: 0s - loss: 0.1882 - ac
curacy: 0.9298
Epoch 2: accuracy improved from 0.78708 to 0.92981, saving model to mode
ls\LSTM.h5
1300/1300 [==============================] - 707s 544ms/step - loss: 0.1
882 - accuracy: 0.9298 - val_loss: 0.2889 - val_accuracy: 0.8880
Epoch 3/6
1300/1300 [==============================] - ETA: 0s - loss: 0.0871 - ac
curacy: 0.9716
Epoch 3: accuracy improved from 0.92981 to 0.97158, saving model to mode
ls\LSTM.h5
1300/1300 [==============================] - 834s 641ms/step - loss: 0.0
871 - accuracy: 0.9716 - val_loss: 0.3048 - val_accuracy: 0.9006
Epoch 4/6
1300/1300 [==============================] - ETA: 0s - loss: 0.0451 - ac
curacy: 0.9853
Epoch 4: accuracy improved from 0.97158 to 0.98531, saving model to mode
ls\LSTM.h5
1300/1300 [==============================] - 902s 694ms/step - loss: 0.0
451 - accuracy: 0.9853 - val_loss: 0.4040 - val_accuracy: 0.8922
Epoch 5/6
1300/1300 [==============================] - ETA: 0s - loss: 0.0220 - ac
curacy: 0.9935
Epoch 5: accuracy improved from 0.98531 to 0.99354, saving model to mode
ls\LSTM.h5
1300/1300 [==============================] - 801s 616ms/step - loss: 0.0
220 - accuracy: 0.9935 - val_loss: 0.4108 - val_accuracy: 0.9106
Epoch 6/6
1300/1300 [==============================] - ETA: 0s - loss: 0.0160 - ac
curacy: 0.9953
Epoch 6: accuracy improved from 0.99354 to 0.99527, saving model to mode
ls\LSTM.h5
1300/1300 [==============================] - 775s 596ms/step - loss: 0.0
160 - accuracy: 0.9953 - val_loss: 0.4495 - val_accuracy: 0.9083
```

In [36]:

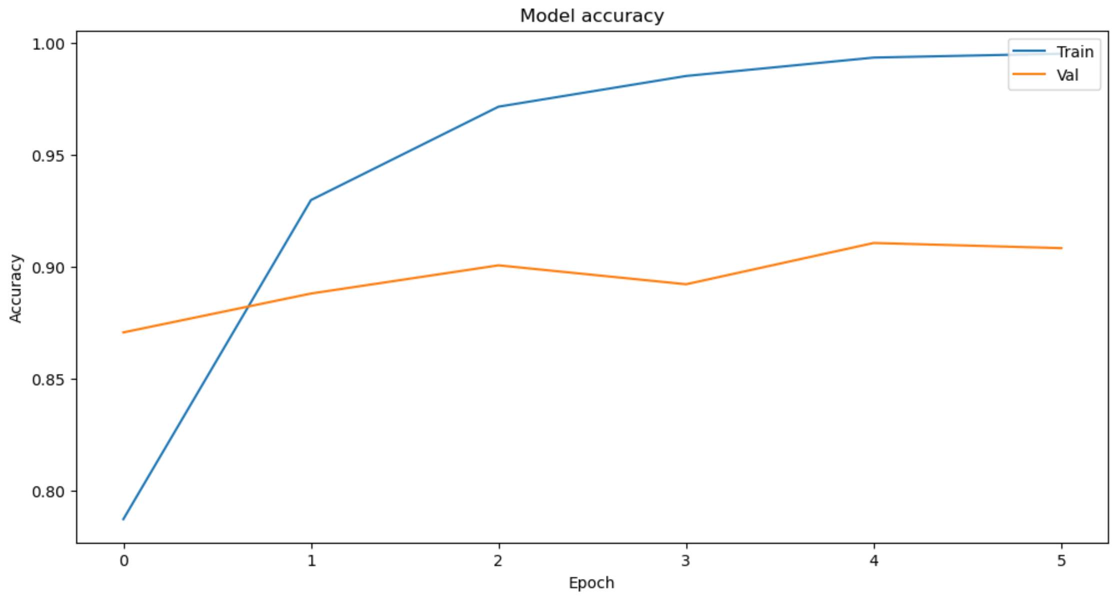
```
plt.figure(figsize=(12,6))
plt.plot(model.history.history['loss'][:])
plt.plot(model.history.history['val_loss'][:])
plt.title('Model loss')
plt.xlabel('epoch')
plt.ylabel('loss (categorical_crossentropy)')
plt.legend(['Train', 'Val'], loc='upper right')
```

Out[36]: <matplotlib.legend.Legend at 0x238c30a1450>



In [37]:

```
plt.figure(figsize=(12,6))
plt.plot(model.history.history['accuracy'][:])
plt.plot(model.history.history['val_accuracy'][:])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```




In [38]:

```
#I grab a random review from the dataframe
value = 200
df_reviews.iloc[value]
```

Out[38]:

```
ProductId                                                          B0028C4
4Z0
ProfileName
Lou
Score
2
Text                          Even with small containers, they don't fill t
h...
HelpfulnessNumerator
0
HelpfulnessDenominator
0
Percent
0.0
Polarity_Rating                                                      Negat
ive
Name: 200, dtype: object
```

In [39]:
```
df_reviews.iloc[value]['Text']
```

Out[39]:
```
"Even with small containers, they don't fill them up.  These little tins
are less than half filled and at the price charged it seems a rip-off. I
s there some exotic ingredient as costly as gold contained in those tiny
squares?  Or how about the cereal ploy, they were filled at the factory
but settled in transport.<br />Can manufacturers be honest in their deal
ings?"
```

In [40]:
```
review = df_reviews.iloc[value]['Text']

regex = re.compile(r'[^a-zA-Z\s]')
review = regex.sub('', review)

words = review.split(' ')
filtered = [w for w in words if w not in english_stops]
filtered = ' '.join(filtered)
filtered = [filtered.lower()]
```

In [41]:
```
tokenize_words = token.texts_to_sequences(filtered)
tokenize_words = pad_sequences(tokenize_words, maxlen=max_length, padding=
```

In [42]:
```
result = model.predict(tokenize_words)

if result >= .50:
    print('Postive')
else:
    print('Negative')
```

```
1/1 [==============================] - 1s 631ms/step
Negative
```

In [43]:
```
#Making an Unique Numeric ID for all the users
df_reviews['UserID'] = pd.factorize(df_reviews['ProfileName'])[0] + 1
```

In [44]:

```
df_reviews[df_reviews['UserID'] == 151].head()
```

Out[44]:

| | ProductId | ProfileName | Score | Text | HelpfulnessNumerator | HelpfulnessDen |
|---|---|---|---|---|---|---|
| **151** | B00374XSVY | Chris | 5 | Works with chicken fish beef or pork. Fast eas... | 0 | |
| **637** | B000G6RYNE | Chris | 4 | This kettle chips taste "Good , Crispy & Crunc... | 0 | |
| **729** | B008BEGP9W | Chris | 4 | I really like the pineapple shortcakes sold he... | 0 | |
| **1006** | B002XG21MO | Chris | 5 | These are just like the animal crackers we eat... | 0 | |
| **3142** | B000FDKQCY | Chris | 5 | This product is as good as any that I have eve... | 0 | |

In [45]:

```
#I had to clean the data some more, some review went back and review the s
cleaned = df_reviews.drop_duplicates(subset=['ProductId', 'UserID'], keep=

print('Removed Dulipcated, Total Reviews {}'.format(cleaned.shape[0]))
print('Total Reviews {}'.format(df_reviews.shape[0]))
```

```
Removed Dulipcated, Total Reviews 49283
Total Reviews 50000
```

In [46]:

```
cleaned.head()
```

Out[46]:

| | ProductId | ProfileName | Score | Text | HelpfulnessNumerator | HelpfulnessDenomin |
|---|---|---|---|---|---|---|
| 0 | B001E4KFG0 | delmartian | 5 | I have bought several of the Vitality canned d... | 1 | |
| 1 | B00813GRG4 | dll pa | 1 | Product arrived labeled as Jumbo Salted Peanut... | 0 | |
| 2 | B000LQOCH0 | Natalia Corres "Natalia Corres" | 4 | This is a confection that has been around a fe... | 1 | |
| 3 | B000UA0QIQ | Karl | 2 | If you are looking for the secret ingredient i... | 3 | |
| 4 | B006K2ZZ7K | Michael D. Bigham "M. Wassir" | 5 | Great taffy at a great price. There was a wid... | 0 | |

In [47]:

```
user_rating_df = cleaned.pivot(index='UserID', columns='ProductId', values
```

In [48]:

```
user_rating_df.head(5)
```

Out[48]:

| ProductId | 2734888454 | B00002NCJC | B00002Z754 | B00005V3DC | B000084DVR | B000084E1U |
|---|---|---|---|---|---|---|
| UserID | | | | | | |
| 0 | NaN | NaN | NaN | NaN | NaN | NaN |
| 1 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2 | NaN | NaN | NaN | NaN | NaN | NaN |
| 3 | NaN | NaN | NaN | NaN | NaN | NaN |
| 4 | NaN | NaN | NaN | NaN | NaN | NaN |

5 rows × 6115 columns

In [49]:
```
tf.compat.v1.disable_eager_execution()
```

In [50]:
```
norm_user_rating_df = user_rating_df.fillna(0) / 5.0
trX = norm_user_rating_df.values
trX[0:5]
```

Out[50]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [51]:

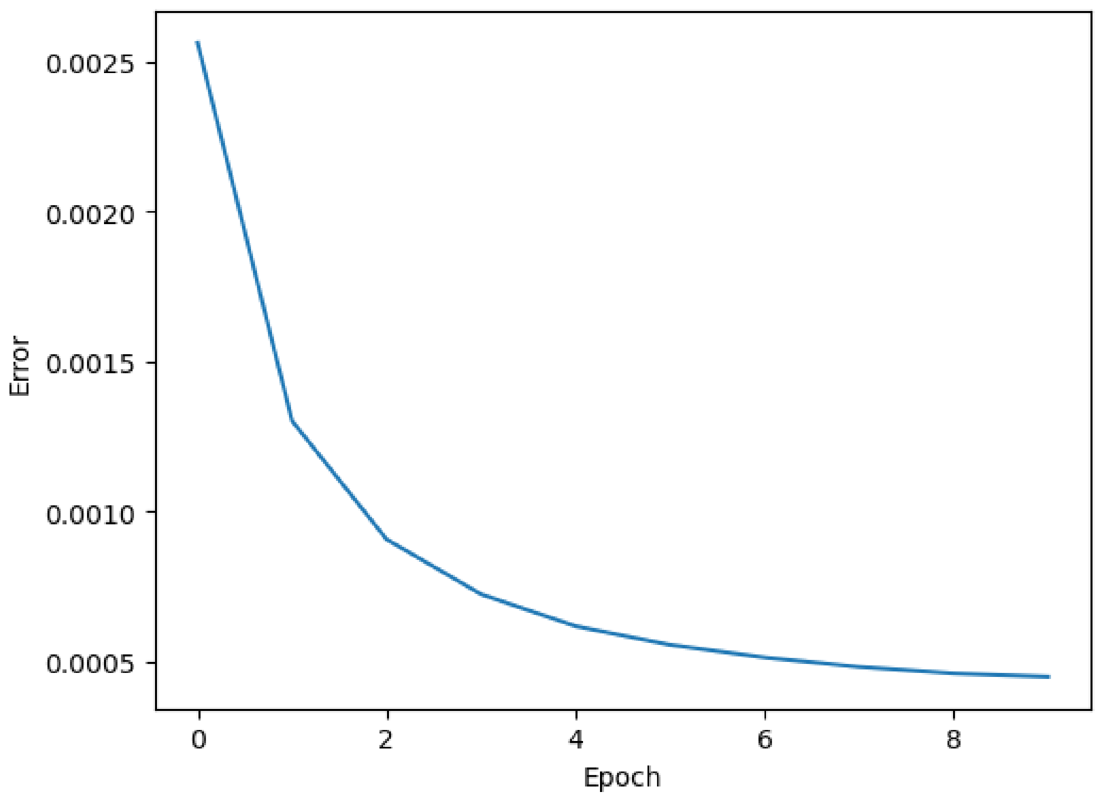
```
hiddenUnits = 20
visibleUnits =  len(user_rating_df.columns)
vb = tf.compat.v1.placeholder("float", [visibleUnits])
hb = tf.compat.v1.placeholder("float", [hiddenUnits])
W = tf.compat.v1.placeholder("float", [visibleUnits, hiddenUnits])

# Process phase 1 of a RBM, use v0, _h0, h0
v0 = tf.compat.v1.placeholder("float", [None, visibleUnits])
_h0 = tf.nn.sigmoid(tf.matmul(v0, W) + hb)
h0 = tf.nn.relu(tf.sign(_h0 - tf.random.uniform(tf.shape(input=_h0))))

#Phase 2: Reconstruction
_v1 = tf.nn.sigmoid(tf.matmul(h0, tf.transpose(W)) + vb)
v1 = tf.nn.relu(tf.sign(_v1 - tf.random.uniform(tf.shape(input=_v1))))
h1 = tf.nn.sigmoid(tf.matmul(v1, W) + hb)

#Learning rate
alpha = 1.0

w_pos_grad = tf.matmul(tf.transpose(v0), h0)
w_neg_grad = tf.matmul(tf.transpose(v1), h1)

CD = (w_pos_grad - w_neg_grad) / tf.compat.v1.to_float(tf.shape(v0)[0])

update_w = W + alpha * CD
update_vb = vb + alpha * tf.reduce_mean(v0 - v1, 0)
update_hb = hb + alpha * tf.reduce_mean(h0 - h1, 0)

err = v0 - v1
err_sum = tf.reduce_mean(err * err)

#Current weight
cur_w = np.zeros([visibleUnits, hiddenUnits], np.float32)

cur_vb = np.zeros([visibleUnits], np.float32)

cur_hb = np.zeros([hiddenUnits], np.float32)

prv_w = np.zeros([visibleUnits, hiddenUnits], np.float32)

prv_vb = np.zeros([visibleUnits], np.float32)

prv_hb = np.zeros([hiddenUnits], np.float32)

sess = tf.compat.v1.Session()
sess.run(tf.compat.v1.global_variables_initializer())


# Using a for loop, Run the model, with 10 epochs, batchsize = 100, and er
epochs = 10
batchsize = 100
errors = []
for i in range(epochs):
    for start, end in zip( range(0, len(trX), batchsize), range(batchsize,
        batch = trX[start:end]
        cur_w = sess.run(update_w, feed_dict={v0: batch, W: prv_w, vb: prv
        cur_vb = sess.run(update_vb, feed_dict={v0: batch, W: prv_w, vb: p
```

```
        cur_nb = sess.run(update_hb, feed_dict={v0: batch, W: prv_w, vb: p
        prv_w = cur_w
        prv_vb = cur_vb
        prv_hb = cur_hb
    errors.append(sess.run(err_sum, feed_dict={v0: trX, W: cur_w, vb: cur_
    print (errors[-1])
```

```
WARNING:tensorflow:From C:\Anaconda\Lib\site-packages\tensorflow\python
\util\dispatch.py:1260: to_float (from tensorflow.python.ops.math_ops) i
s deprecated and will be removed in a future version.
Instructions for updating:
Use `tf.cast` instead.
0.0025635296
0.0013016531
0.0009061099
0.0007227106
0.00061612227
0.00055364566
0.00051163853
0.00048007743
0.00045839624
0.0004471017
```

In [52]:

```
plt.plot(errors)
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.show()
```



In [53]:
```python
mock_user_id = 151
```

In [54]:
```python
inputUser = trX[mock_user_id-1].reshape(1, -1)
inputUser[0:5]
```

Out[54]:
```
array([[0., 0., 0., ..., 0., 0., 0.]])
```

In [55]:
```python
# Feeding in the user and reconstructing the input. use sigmoid
hh0 = tf.nn.sigmoid(tf.matmul(v0, W) + hb)
vv1 = tf.nn.sigmoid(tf.matmul(hh0, tf.transpose(W)) + vb)
feed = sess.run(hh0, feed_dict={ v0: inputUser, W: prv_w, hb: prv_hb})
rec = sess.run(vv1, feed_dict={ hh0: feed, W: prv_w, vb: prv_vb})
```

In [56]:
```python
scored_asin_df_mock = cleaned.drop_duplicates(subset=['ProductId'])
scored_asin_df_mock = scored_asin_df_mock.assign(RecommendationScore = rec
scored_asin_df_mock[['ProductId','RecommendationScore']].sort_values(["Rec
```

Out[56]:

| | ProductId | RecommendationScore |
|---|---|---|
| **4063** | B001EW5YQS | 0.025583 |
| **7321** | B0042395CA | 0.012464 |
| **45379** | B000FVVZ3U | 0.008445 |
| **40981** | B000IKEGRK | 0.005815 |
| **36090** | B002DHN1UE | 0.005210 |
| **17705** | B0010AR1E2 | 0.004573 |
| **5012** | B000HDKZDC | 0.004495 |
| **17240** | B000I6PZVK | 0.004260 |
| **14515** | B0016814QO | 0.004161 |
| **33278** | B001RQAVKK | 0.004146 |
| **29077** | B000NU8H0C | 0.004062 |
| **2950** | B003XKKEBE | 0.003626 |
| **18894** | B003FSPWXY | 0.003173 |
| **4701** | B0048IPR9Y | 0.002770 |
| **17127** | B008FSIHNG | 0.002756 |
| **13021** | B001EPQ8XS | 0.002741 |
| **35800** | B00290W1CY | 0.002572 |
| **13806** | B000FFIIYU | 0.002498 |
| **19006** | B003R0LM48 | 0.002466 |
| **7881** | B000VZJJIS | 0.002433 |

In [57]:
```
food_df_mock = cleaned[cleaned['UserID'] == 151]
food_df_mock.head()
```

Out[57]:

| | ProductId | ProfileName | Score | Text | HelpfulnessNumerator | HelpfulnessDen |
|---|---|---|---|---|---|---|
| **151** | B00374XSVY | Chris | 5 | Works with chicken fish beef or pork. Fast eas... | 0 | |
| **637** | B000G6RYNE | Chris | 4 | This kettle chips taste "Good , Crispy & Crunc... | 0 | |
| **729** | B008BEGP9W | Chris | 4 | I really like the pineapple shortcakes sold he... | 0 | |
| **1006** | B002XG21MO | Chris | 5 | These are just like the animal crackers we eat... | 0 | |
| **3142** | B000FDKQCY | Chris | 5 | This product is as good as any that I have eve... | 0 | |

In [58]:
```
merged_df_mock = scored_asin_df_mock.merge(food_df_mock, on='ProductId', h
```

In [59]:
```
merged_df_mock[['ProductId','RecommendationScore']].sort_values(["Recommen
```

Out[59]:

| | ProductId | RecommendationScore |
|---|---|---|
| **649** | B001EW5YQS | 0.025583 |
| **1122** | B0042395CA | 0.012464 |
| **5653** | B000FVVZ3U | 0.008445 |
| **5023** | B000IKEGRK | 0.005815 |
| **4366** | B002DHN1UE | 0.005210 |

Above are the top 5 items that Chris will be interested in purchasing for reviewing.

## Conclusion

**In this project, I successfully conducted sentiment analysis on the Amazon Fine Food Reviews dataset using Jupyter Notebooks and various natural language processing techniques. Our main objective was to recommend new products to someone who does reviews often (Chris), using RNN to perform sentiment analysis using only Amazon reviews as the model. I began by performing exploratory data analysis to gain insights into the dataset's characteristics and distribution of sentiments. Preprocessing steps, such as removing stop words, were applied to clean the text data and enhance the quality of our sentiment analysis. This analysis provided valuable insights into the key factors driving positive or negative sentiment within the Amazon Fine Food Reviews dataset.**

**The Sentiment Analysis on Amazon Fine Food Reviews project successfully employed deep learning techniques to classify customer reviews as positive or negative. The outcome of this project contributes to a better understanding of customer sentiments and can be potentially applied in various industries for customer feedback analysis and decision-making processes. Overall, this project was a success.**